IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	: FULTHEIM et al.	
Serial No.	: 10/828,465	Group Art Unit: 2128
Filed	: April 21, 2004	Examiner: David Silver
For	: CLUSTER BASED OPERATING SYSTEM-AGNOSTIC VIRTUAL COMPUTING SYSTEM	

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

APPEAL BRIEF

(1) Real Party in Interest

The subject application is owned by ScaleMP Inc., having a place of business at 20863 Stevens Creek Blvd., Cupertino, California 94087. The assignment was recorded in the U.S.P.T.O. on April 21, 2004, under Reel 015253, Frame 0977.

(2) Related Appeals and Interferences

None.

(3) Status of Claims

This application as filed contained claims 1-37. Claims 38-45 were added by amendment. Claims 1-45 are currently pending and were finally rejected in an Official Action dated July 16, 2010.

On September 16, 2010, Appellant appealed from the rejection of claims 1-45 (all the claims currently pending in this application). A pre-appeal brief request for review was submitted together with the notice of appeal. On January 6, 2011, the conference panel issued a decision that the appeal should proceed to the Board of Patent Appeals and Interferences.

(4) Status of Amendments

No amendments have been made since the Official Action of July 16, 2010.

(5) Summary of Claimed Subject Matter

One aspect of Appellant's invention, as recited in **independent claim 1**, provides a method for executing a software application in a plurality of computers, as illustrated by computing nodes 22, 24, 26 in Fig. 1 and by similar sorts of arrangements in Figs. 4-6. The computers have hardware resources that include memory 30 and I/O devices 32 (page 15, lines 4-7) and intercommunicate over a network 44.

The method includes the following steps:

a) At least first and second computers run respective virtual machine implementers, such as virtual machine monitors (VMM) 34, 36 in Fig. 1. (Page 10, lines 11-14, notes that the virtual machine implementers may be virtual machine monitors.) The virtual machine implementers run separately and independently of one another, as explained, for example, on page 15, lines 20-22: "…virtual machine monitors 34, 36… can differ in implementation technique or hardware. For example, the virtual machine monitors 34, 36 could be different products…"; as well as on page 16, lines 20-22: "The system 10 can be constructed using different types of emulators and different types of virtual machine monitors in many combinations."

b) A virtual machine executes on the computers, and is shared between the virtual machine implementers, as shown in Figs. 1 and 6, for example (VM 20). The execution of the virtual machine uses the respective I/O devices in the computers for intercommunication, as explained on page 16, line 31 – page 17, line 12. A guest operating system runs over the shared virtual machine, as illustrated by GUEST OS 18 in Fig. 1 and described on page 16, lines 1-3.

Independent claim 14 recites a computer software product, which comprises a computer-readable medium, as described on page 13, lines 11-23. The instructions cause a plurality of computers to perform a method for executing a software application, as illustrated by computing nodes 22, 24, 26 in Fig. 1 and by similar sorts of arrangements in Figs. 4-6. The computers have hardware resources that include memory 30 and I/O devices 32 (page 15, lines 4-7) and intercommunicate over a

network 44. The steps in the method performed by the computers are similar to those in claim 1 and are repeated here for the sake of completeness:

a) At least first and second computers run respective virtual machine implementers, such as virtual machine monitors (VMM) 34, 36 in Fig. 1. The virtual machine implementers run separately and independently of one another, as explained, for example, on page 15, lines 20-22: "…virtual machine monitors 34, 36… can differ in implementation technique or hardware. For example, the virtual machine monitors 34, 36 could be different products…"; as well as on page 16, lines 20-22: "The system 10 can be constructed using different types of emulators and different types of virtual machine monitors in many combinations."

b) A virtual machine executes on the computers, and is shared between the virtual machine implementers, as shown in Figs. 1 and 6, for example (VM 20). The execution of the virtual machine uses the respective I/O devices in the computers for intercommunication, as explained on page 16, line 31 – page 17, line 12. A guest operating system runs over the shared virtual machine, as illustrated by GUEST OS 18 in Fig. 1 and described on page 16, lines 1-3.

Independent claim 28 recites a computer system for executing a software application. The system includes a plurality of computers, as illustrated by computing nodes 22, 24, 26 in Fig. 1 and by similar sorts of arrangements in Figs. 4-6. The computers have hardware resources that include memory 30 and I/O devices 32 (page 15, lines 4-7) and intercommunicate over a network 44 that is connected to the computers.

At least first and second computers execute virtual machine implementers, such as virtual machine monitors (VMM) 34, 36 in Fig. 1. The virtual machine implementers run separately and independently of one another, as explained, for example, on page 15, lines 20-22: "…virtual machine monitors 34, 36… can differ in implementation technique or hardware. For example, the virtual machine monitors 34, 36 could be different products…"; as well as on page 16, lines 20-22: "The system 10 can be constructed using different types of emulators and different types of virtual machine monitors in many combinations." A virtual machine is implemented concurrently and

shared by the virtual machine implementers, as shown in Figs. 1 and 6, for example (VM 20).

The computers execute a guest operating system over the shared virtual machine, as illustrated by GUEST OS 18 in Fig. 1, and the software application (applications 12, 14, 16) executes over the guest operating system, as described on page 14, lines 12-17, and page 16, lines 12-16. Commands invoked by the software application are received the by virtual machine implementers, as explained on page 9, lines 4-10. The hardware resources are shared by communicating over the network using respective I/O devices in the computers, as explained on page 16, line 31 – page 17, line 12.

The remaining claims in the application depend from the above independent claims.

(6) Grounds of Rejection to be Reviewed on Appeal

Claims 1-4, 9-16, 22-30 and 32-45 were rejected under 35 U.S.C. 103(a) over Okamoto (U.S. Patent 5,829,041) in view of the *VMware Workstation User's Manual* Version 3.2 (hereinafter "VMware"), while claims 5-8, 17-21 and 31 were rejected over Okamoto and VMware in view of Altman et al. (U.S. Patent Application Publication 2004/0054517).

Appellant believes the rejection of claims 1-45 should be reversed.

(7) Argument

I. Objective Evidence of Non-obviousness

A. Evidence submitted by Appellant stands unchallenged

On October 19, 2009, Appellant submitted declarations under 37 CFR 1.132 by Dr. Joseph Landman and by Boaz Yehuda demonstrating that the invention recited in the claims of the present patent application is objectively non-obvious on a number of grounds. These declarations and supporting exhibits are listed and submitted herewith in Appendix B.

The declarants related to the vSMP product that is sold by the Appellant, ScaleMP Inc., and embodies the claimed invention. They explained that the claimed

invention yields unexpected results and satisfies a long-felt need in the High-Power Computing (HPC) industry, and has also led to commercial success and imitation. Dr. Landman's declaration on this point is supported by exhibits containing documentary evidence (see paragraphs 14-18 in the declaration and Exhibits B-E).

Specifically, Dr. Landman stated as follows (paragraph 15):

> "vSMP addresses a need that has long been felt in the field of high-performance computing (HPC): How to achieve the computing performance level of a supercomputer without high-cost dedicated hardware and special-purpose software? vSMP solves this problem by enabling multiple, generic, low-cost computers to be coupled together efficiently to create a single (virtual) high-power machine, in the manner that is described and claimed in the Application. Until very recently, VMWare virtualization was not even used in HPC, and its use in this area is still very limited due to the overhead of the virtualization process. vSMP has overcome these limitations and is rapidly growing in use and recognition."

He went on to point out that vSMP has been adopted by industry leaders, including Hewlett-Packard, Dell, IBM, Sun Microsystems, Cray, and Silicon Graphics, and that at least one imitative product has now entered the market.

Mr. Yehuda provided testimony based on his own experience as a user of vSMP during his period as a Country Manager for Sun Microsystems. His declaration attests to his own initial skepticism regarding the workability of the vSMP technology and his understanding of the long-felt need met by the technology.

In the Official Action issued December 30, 2009, following the submission of these declarations, <u>the Examiner did not consider – and in fact did not even mention – the declarations and the secondary considerations that they raised</u>.

Appellant pointed out this lacuna in the response to this Official Action, which was filed April 26, 2010. This response was accompanied by a declaration under 37 CFR 1.132 by Dr. Guy Tel-Zur (attached hereto in Appendix B). Dr. Tel-Zur related mainly to the reasons why a person of ordinary skill could not have derived the claimed

invention from the cited references, but he also endorsed Dr. Landman's finding that the invention claimed in the present patent application is a surprising result, which satisfies a long-felt need (paragraph 17).

Yet in the Final Official Action of July 16, 2010, the Examiner again failed to relate to the substance of the secondary considerations raised by the declarants and simply dismissed the evidence out of hand:

> "15. Consideration was indeed given to the previously submitted secondary consideration but the Declaration was unpersuasive. Specifically, it pointed out similar issues as above, and has thus been addressed explicitly herewithin."

The section "above" in the Examiner's remarks relates to Dr. Tel-Zur's explanation of why the references cited in the Official Action do not support a *prima facie* case of obviousness. This section says nothing about the long-felt need for the invention, its commercial success, or any other secondary considerations. The Examiner stated no reason at all for his conclusion that the secondary considerations raised by the declarants were "unpersuasive," nor did he ever relate explicitly to the testimony provided by Dr. Landman and Mr. Yehuda.

Thus, over two rounds of prosecution, <u>the Examiner did not refute the substance of the secondary considerations that Appellant introduced as objective evidence of non-obviousness</u>. Under these circumstances, the objective evidence of non-obviousness must be taken at face value as uncontroverted.

B. Secondary considerations cannot be disregarded

It is very clear that the Examiner has failed to meet the requirements of MPEP 716.01(B) with respect to the objective evidence of non-obviousness raised in the Declarations by Landman and Yehuda:

> *Consideration of evidence*. Evidence traversing rejections, when timely presented, must be considered by the examiner whenever present. All entered affidavits, declarations, and other evidence traversing rejections are

> acknowledged and commented upon by the examiner in the next succeeding action… Where the evidence is insufficient to overcome the rejection, the examiner must specifically explain why the evidence is insufficient. General statements such as "the declaration lacks technical validity" or "the evidence is not commensurate with the scope of the claims" without an explanation supporting such findings are insufficient.

The Examiner's conclusory statement, quoted above, that "the Declaration was unpersuasive" is plainly within the bounds of what MPEP considers to be "insufficient" in this regard.

MPEP 716.01(a) continues:

> OBJECTIVE EVIDENCE MUST BE CONSIDERED WHEN TIMELY PRESENT
>
> Affidavits or declarations, when timely presented, containing evidence of criticality or unexpected results, commercial success, long-felt but unsolved needs, failure of others, skepticism of experts, etc., must be considered by the examiner in determining the issue of obviousness of claims for patentability under 35 U.S.C. 103. The Court of Appeals for the Federal Circuit stated in *Stratoflex, Inc. v. Aeroquip Corp.*, 713 F.2d 1530, 1538, 218 USPQ 871, 879 (Fed. Cir. 1983) that "evidence rising out of the so-called 'secondary considerations' must always when present be considered en route to a determination of obviousness."

The Examiner has not commented on (and has barely even acknowledged) the declarations by Dr. Landman and Mr. Yehuda, and certainly has not given an explanation to support his findings regarding the evidence of secondary considerations provided by these declarations.

The rules stated in MPEP are supported by well-established case law. Most recently, the Court of Appeals for the Federal Circuit overturned a summary judgment by the district court for failure to consider objective evidence of non-obviousness in

Transocean Offshore Deepwater Drilling, Inc. v. Maersk Contractors USA, Inc., 617 F.3d 1296, 1305 (Fed. Cir. 2010), (citations omitted):

> We hold that the district court erred by failing to consider Transocean's objective evidence of nonobviousness. Our case law is clear that this type of evidence "must be considered in evaluating the obviousness of a claimed invention"... While it is true that we have held in individual cases that objective evidence of nonobviousness did not overcome the strong *prima facie* case – this is a case-by-case determination... To be clear, a district court must always consider any objective evidence of non-obviousness presented in a case... (page 11)

Although the Examiner in the present application appears to have taken the position that he may consider or disregard Appellant's objective evidence of non-obviousness at his option, this position is in outright contradiction to the dictates of MPEP and the case law.

C. Conclusion: All the claims in the application are non-obvious

MPEP and the case law make clear that objective evidence of non-obviousness (secondary considerations) submitted by the applicant must be considered and given due weight by the Patent Office. In the present case, however, the Examiner has chosen not to give Appellant's evidence any weight at all.

Proper consideration of the evidence that Appellant has submitted in the present case leads to a clear conclusion that the claimed invention is non-obvious. There is no evidence on the record, nor even a reasoned explanation, that could be taken in any way to refute Appellant's evidence and arguments in this regard. Even putting aside the question of *prima facie* obviousness (which Appellant does not concede by any means), the rejection of the claims in this application should be overturned because they are objectively non-obvious, and the application should therefore be allowed.

II. Prima Facie Non-obviousness of the Claims

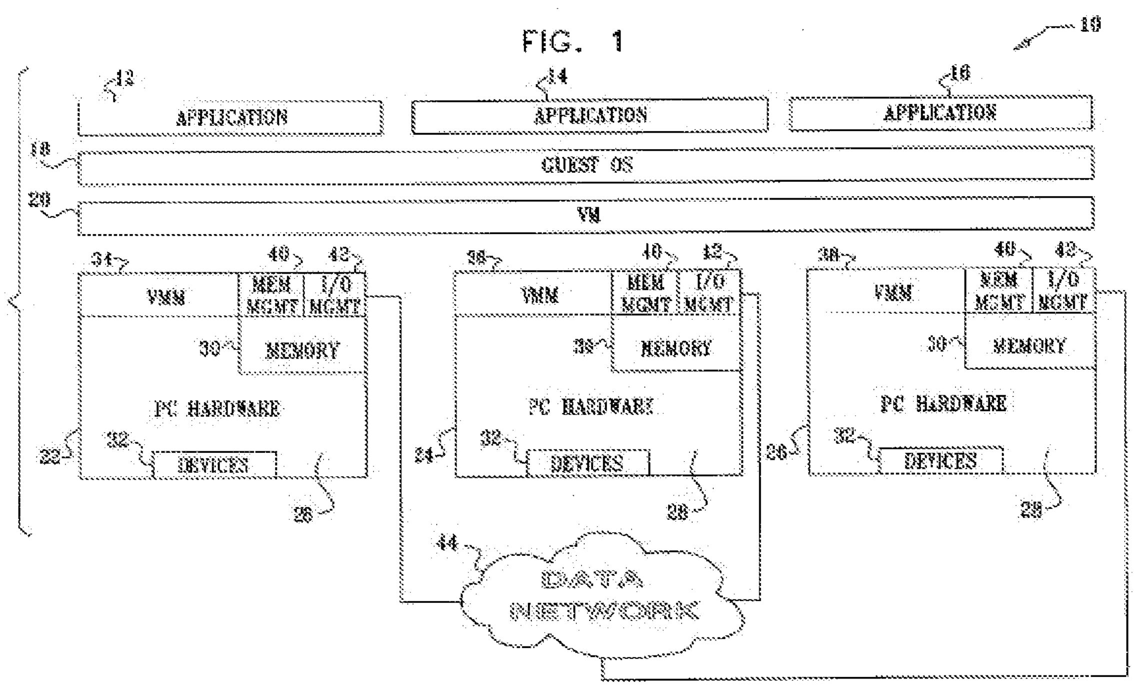
Notwithstanding the objective evidence of non-obviousness that Appellant has submitted, the Examiner has failed to make a *prima facie* case of obviousness against the claims. Although the evidence reviewed above is believed to be sufficient to prove that the claims are non-obvious, Appellant will show in addition that a person of ordinary skill in the art would not and could not have created the claimed invention on the basis of the cited references.

A. The Section 103(a) rejection of independent claims 1, 14 and 28

Appellant respectfully submits that the Examiner erred in maintaining that claims 1, 14 and 28 are obvious over Okamoto in view of VMware.

The claimed invention is well represented by Fig. 1 of the present patent application, which is reproduced below:

FIG. 1



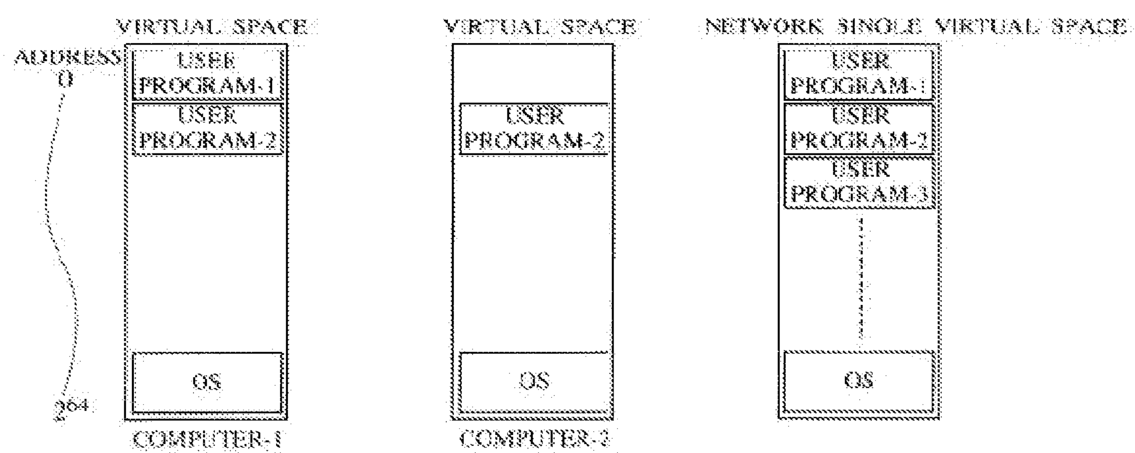
The key point, as explained at length in the attached declarations by Landman, Tel-Zur and Eylon, is that a <u>virtual machine is shared between multiple computers</u>. The virtual

machine (shown as VM 20 in the above figure) is supported by separate and independent virtual machine implementers on the different computers (shown as virtual machine monitors - VMMs 34, 36, 38 - on computers 22, 24, 26). A guest operating system (GUEST OS 18) runs over the shared virtual machine, and software applications (APPLICATION 12, 14, 16) execute on the guest operating system.

In other words, as explained by the declarants, the shared virtual machine appears to the guest operating system and the application software as if it was a single, high-powered computer, even though it uses the resources of multiple physical computers. The term "virtual machine" is a term of art, which is defined as follows in the VMware reference that was cited by the Examiner: A virtual machine "is equivalent to a PC with a unique network address and a full complement of hardware devices" (emphasis added).

For comparison, we now present the system shown by Okamoto:

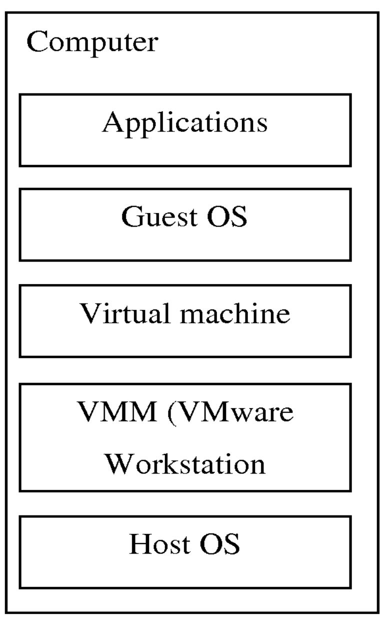
FIG. 2



The computers are connected by a network (as shown in Fig. 1). Each computer has an operating system (OS) and runs software applications (shown as "user programs" in the figure above. One of the computers, shown at the right in the figures, acts as a "chapter server" and manages a "virtual [memory] space management scheme" within the system. Each computer, however, manages its part of the virtual space independently (abstract).

Comparing Okamoto's figures and description to claim 1 and to Fig. 1 shown on the preceding page, striking differences immediately become evident: Okamoto has no virtual machine at all – not in the sense in which the term is understood in the art or defined by the very VMware reference that the Examiner himself cited. In the absence of any sort of virtual machine, it is clear that Okamoto cannot and does not teach or suggest a virtual machine shared among his multiple physical computers, as illustrated in Fig. 1. Furthermore, in the absence of a virtual machine, Okamoto has no use for virtual machine implementers, nor can he support a guest operating system, both of which are shown in Fig. 1 and are required by claim 1. In the final analysis, the only commonality between Okamoto and the claimed invention is that both have the elements recited in the claim preamble: a plurality of computers connected by a network executing a software application, along with the unfortunate semantic coincidence of using the same term "virtual" to mean different things.

The VMware reference describes a virtual machine implementer, referred to as the "VMware Workstation." According to the instructions provided in the reference (pages 13-15), the implementer runs on a single computer over the host operating system, and supports one or more virtual machines with guest operating systems. These points are further explained in paragraphs 14-15 of the Tel-Zur declaration. The model of operation is shown by the diagram below:



This layered model is similar to that shown in Fig. 1 above, except that <u>there is no hint in VMware that the virtual machines might possibly be shared among multiple virtual machine implementers or multiple computers</u>. All of the expert declarations that Appellant has submitted make clear that despite the success and wide adoption of VMware products, it was not considered feasible to share virtual machines in this way prior to the present invention.

The Examiner asserted in the Final Official Action (page 7, paragraph 16) that "Nothing in Okamoto or VMware prevents the virtual machine from working on Okamoto' s system." This statement is true, up to a point: A person of ordinary skill in the art who was familiar with Okamoto and VMware might well have installed the VMware Workstation on each of Okamoto's separate computers, and then would have been able to run Okamoto's user programs over virtual machines and guest operating systems <u>on each computer individually</u>. This is the result that the person of ordinary skill would have obtained.

Without the benefit of hindsight from the present patent application, however, the person of ordinary skill would not even have considered it possible to share a virtual machine over multiple virtual machine implementers and computers. As noted earlier, there is no suggestion in the cited references that a virtual machine could run in this manner. As explained by Dr. Tel-Zur (paragraph 15), the person of ordinary skill would have understood that the VMware virtual machine is not just a distributed application program and could not be distributed over multiple computers in any simple way. Dr. Landman provides (paragraph 11) a tabular summary of the changes that would be needed to extend the VMware Workstation to enable shared virtual machine implementation. None of the cited references explains how these changes could be effected. The person of ordinary skill, even if motivated, would therefore have had no expectation of success in attempting to combine the teachings of Okamoto and VMware to create the invention claimed in the present patent application.

The Supreme Court decision in KSR International Co. v. Teleflex Inc. et al, 550 U.S. 398 (2007) noted with approval In re Kahn, 441 F. 3d 977, 988 (CA Fed. 2006), which stated that "rejections on obviousness grounds cannot be sustained by mere conclusory statements..." As shown above, however, the Examiner's finding that

claims 1, 14 and 28 are obvious over Okamoto and VMware is based entirely on hindsight and his own conclusory statements, without any support of substance in the cited art. The Examiner has therefore failed to make a *prima facie* case of obviousness against claims 1, 14 and 28, and the rejection of these claims should be reversed.

B. The Section 103(a) Rejection of Claims 38 and 39

Appellant respectfully submits that even if the independent claims in this application were conceded to be obvious over the cited references, these references still do not teach or suggest the added elements of dependent claims 38 and 39.

Claims 38 and 39 depend from independent claims 1 and 14, respectively, and add the limitation that there is exactly one instance of a single guest operating system running over the multiple virtual machine implementers recited in the independent claims. The Examiner gave no specific grounds of rejection for claims 38 and 39, and stated merely the "The Instant Claims recite substantially same limitations as the above-rejected claims [1-4 and 9-13] and are therefore rejected under same prior-art teachings." In fact, however, none of claims 1-4 or 9-13 says anything at all about one instance of a single guest operating system. As noted earlier in reference to the independent claims, both Okamoto and VMware describe multiple operating system instances – at least one per computer.

Therefore, claims 38 and 39 are independently patentable over the cited art, notwithstanding the patentability of claims 1 and 14.

Summary

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-45 was erroneous. Reversal of the Examiner's decision is respectfully requested.

Respectfully submitted,
/DANIEL KLIGLER/
Daniel Kligler, Reg. No.41,120
March 4, 2011

APPENDIX A

1. A method for executing a software application in a plurality of computers having respective hardware resources said hardware resources comprising a respective memory and a respective I/O device, wherein said computers include a first computer and a second computer that intercommunicate over a network, comprising the steps of:

running at least a first virtual machine implementer and a second virtual machine implementer on said first computer and said second computer, using said respective memory, wherein said first and second virtual machine implementers run separately and independently of one another on said first and second computers, respectively; and

executing a virtual machine on said computers, wherein said virtual machine is shared between said first virtual machine implementer and said second virtual machine implementer using said respective I/O device in each of said first computer and said second computer to intercommunicate between said first computer and said second computer, and a guest operating system runs over said shared virtual machine.

2. The method according to claim 1, further comprising the step of running said software application over said guest operating system, so that commands invoked by said software application are monitored or emulated by said first virtual machine implementer and said second virtual machine implementer on said first computer and said second computer, while said hardware resources of said first computer and said second computer are shared by communication over said network.

3. The method according to claim 1, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is a virtual machine monitor.

4. The method according to claim 1, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is an emulator.

5. The method according to claim 1, wherein at least said first computer comprises a first virtual node comprising a first physical CPU of said first computer and a second virtual node comprising a second physical CPU of said first computer.

6. The method according to claim 1, wherein said virtual machine comprises a first virtual machine and a second virtual machine, wherein said first virtual machine and said second virtual machine have a plurality of virtual CPU's that are virtualized by said first virtual machine implementer based on a first physical CPU and said second virtual machine implementer based on a second physical CPU, respectively.

7. The method according to claim 6, and a first virtual node comprises said first physical CPU and said second physical CPU.

8. The method according to claim 7, wherein said first virtual machine implementer virtualizes at least one of said virtual CPU's of said first virtual machine

based on said first physical CPU and virtualizes at least one of said virtual CPU's in said second virtual machine based on said second physical CPU.

9. The method according to claim 1, further comprising the steps of:

providing a management system for said first virtual machine implementer and said second virtual machine implementer to control said first computer and said second computer, respectively, wherein said management system comprises a wrapper for receiving calls to a device driver from said first virtual machine implementer, said wrapper invoking said device driver according to a requirement of said first virtual machine implementer.

10. The method according to claim 9, further comprising the step of providing a virtual PCI controller for said management system to control a physical PCI controller in one of said computers.

11. The method according to claim 9, further comprising the step of providing a virtual DMA controller for said management system to control a physical DMA controller in one of said computers.

12. The method according to claim 11, further comprising the steps of:

providing a virtual PCI controller to control a physical PCI controller in one of said computers; and

during a bootup phase of operation scanning a device list with said virtual PCI controller to identify devices having on-board DMA controllers.

13. The method according to claim 1, further comprising the steps of:

with said first virtual machine implementer and said second virtual machine implementer maintaining mirrors of a portion of said respective memory that is used by said guest operating system in each of said computers;

write-invalidating at least a portion of a page of said respective memory in one of said computers; and

transferring a valid copy of said portion of said page to said one computer from another of said computers via said network.

14. A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions cause a plurality of computers having respective hardware resources, said hardware resources comprising a respective memory and a respective I/O device, to perform a method for executing a software application, wherein said computers include a first computer and a second computer that intercommunicate over a network, comprising the steps of:

running at least a first virtual machine implementer and a second virtual machine implementer on said first computer and said second computer, using said respective memory, wherein said first and second virtual machine implementers run separately and independently of one another on said first and second computers, respectively; and

executing a virtual machine on said computers, wherein said virtual machine is shared between said first virtual machine implementer and said second virtual machine implementer using said respective I/O device in each of said first computer and said second computer to intercommunicate between said first computer and said second computer, and a guest operating system runs over said shared virtual machine.

15. The computer software product according to claim 14, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is a virtual machine monitor.

16. The computer software product according to claim 14, wherein at least one of said first virtual machine implementer and said second virtual machine implementer is an emulator.

17. The computer software product according to claim 14, wherein at least said first computer comprises a first virtual node comprising a first physical CPU of said first computer and a second virtual node comprising a second physical CPU of said first computer.

18. The computer software product according to claim 17, wherein said virtual machine comprises a first virtual machine and a second virtual machine, wherein said first virtual machine and said second virtual machine have a plurality of virtual CPU's that are virtualized by said first virtual machine implementer based on said first physical

CPU and said second virtual machine implementer based on said second physical CPU, respectively.

19. The computer software product according to claim 18, wherein said plurality of virtual CPU's that are virtualized by said first virtual machine implementer based on said first physical CPU and said second virtual machine implementer based on said second physical CPU, respectively.

20. The computer software product according to claim 18, wherein said first virtual node comprises said first physical CPU and said second physical CPU.

21. The computer software product according to claim 20, wherein said first virtual machine implementer virtualizes at least one of said virtual CPU's of said first virtual machine based on said first physical CPU and virtualizes at least one of said virtual CPU's in said second virtual machine based on said second physical CPU.

22. The computer software product according to claim 14, wherein said computer is further instructed to perform the step of running said software application over said guest operating system, so that commands invoked by said software application are received by said first virtual machine implementer and said second virtual machine implementer on said first computer and said second computer, while said hardware resources of said first computer and said second computer are shared by communication over said network.

23. The computer software product according to claim 14, further comprising the steps of:

providing a management system for said first virtual machine implementer and said second virtual machine implementer to control said first computer and said second computer, respectively, wherein said management system comprises a wrapper for receiving calls to a device driver from said first virtual machine implementer and said second virtual machine implementer, said wrapper invoking said device driver according to a requirement of said first virtual machine implementer and said second virtual machine implementer.

24. The computer software product according to claim 23, further comprising the step of providing a virtual PCI controller for said management system to control a physical PCI controller in one of said computers.

25. The computer software product according to claim 23, wherein said computers are further instructed to perform the step of providing a virtual DMA controller for said management system to control a physical DMA controller in one of said computers.

26. The computer software product according to claim 25, wherein said computers are further instructed to perform the steps of:

providing a virtual PCI controller to control a physical PCI controller in one of said computers; and

during a bootup phase of operation scanning a device list with said virtual PCI controller to identify devices having on-board DMA controllers.

27. The computer software product according to claim 14, wherein said computers are further instructed to perform the steps of:

with said first virtual machine implementer and said second virtual machine implementer maintaining mirrors of a portion of said respective memory that is used by said guest operating system in each of said computers;

write-invalidating at least a portion of a page of said respective memory in one of said computers; and

transferring a valid copy of said portion of said page to said one computer from another of said computers via said network.

28. A computer system for executing a software application, comprising:

a plurality of computers having respective hardware resources, said hardware resources comprising a respective memory and a respective I/O device, said computers comprising at least a first computer and a second computer;

a network connected to said first computer and said second computer providing intercommunication therebetween;

said first computer and said second computer being operative to execute a first virtual machine implementer and a second virtual machine implementer respectively

using said respective memory, wherein a virtual machine is implemented concurrently and shared by at least said first virtual machine implementer and said second virtual machine implementer, and wherein said first and second virtual machine implementers run separately and independently of one another on said first and second computers, respectively; and

said computers being operative to execute a guest operating system over said shared virtual machine, wherein said software application executes over said guest operating system, so that commands invoked by said software application are received by said first virtual machine implementer and said second virtual machine implementer on said first computer and said second computer, while said hardware resources of said first computer and said second computer are shared by communication over said network using said respective I/O device.

29. The computer system according to claim 28, wherein said software application comprises a first software application and a second software application, said guest operating system comprises a first guest operating system and a second guest operating system, and said virtual machine comprises a first virtual machine and a second virtual machine, wherein said first software application and said first guest operating system are associated with said first virtual machine, and said second software application and said second guest operating system are associated with said second virtual machine.

30. The computer system according to claim 29, wherein one of said computers has a first physical CPU and a second physical CPU, and said first virtual machine implementer virtualizes a first virtual CPU in said first virtual machine based on said first physical CPU and virtualizes a second virtual CPU in said second virtual machine based on said second physical CPU.

31. The computer system according to claim 28, wherein at least said first computer comprises a first virtual node and a second virtual node.

32. The computer system according to claim 31, wherein said first computer comprises a first processor and a second processor, a first I/O device and a second I/O device, wherein said first I/O device is assigned to said first processor, and said second I/O device is assigned to said second processor.

33. The computer system according to claim 28, further comprising a minimal operating system executing in each of said computers to invoke said first virtual machine implementer and said second virtual machine implementer so that said first virtual machine implementer and said second virtual machine implementer control said computers.

34. The computer system according to claim 28, further comprising a management system for said first virtual machine implementer and said second virtual machine implementer to control said first computer and said second computer,

respectively, wherein said management system comprises a wrapper for receiving calls to a device driver from said first virtual machine implementer and said second virtual machine implementer, said wrapper invoking said device driver according to a requirement of said first virtual machine implementer and said second virtual machine implementer.

35. The computer system according to claim 34, further comprising a virtual PCI controller for said management system to control a physical PCI controller in one of said computers.

36. The computer system according to claim 34, further comprising a virtual DMA controller for said management system to control a physical DMA controller in one of said computers.

37. The computer system according to claim 28, further comprising a memory management system that maintains mirrors of a portion of said respective memory that is used by said guest operating system in each of said computers, wherein said memory management system write-invalidates at least a portion of a page of said respective memory in one of said computers; and transfers a valid copy of said portion of said page to said one computer from another of said computers via said network.

38. The method according to claim 1, wherein said guest operating system consists of exactly one instance of a single guest operating system.

39. The computer software product according to claim 14, wherein said guest operating system consists of exactly one instance of a single guest operating system.

40. The method according to claim 1, wherein said first virtual machine implementer and said second virtual machine implementer are operative to present said respective memory of said first computer and said respective memory of said second computer as a single shared memory to said guest operating system, the method further comprising the step of distributing instructions of said guest operating system to said single shared memory.

41. The computer software product according to claim 14, wherein said first virtual machine implementer and said second virtual machine implementer are operative to present said respective memory of said first computer and said respective memory of said second computer as a single shared memory to said guest operating system, and to distribute instructions of said guest operating system to said single shared memory.

42. The computer system according to claim 28, wherein said first virtual machine implementer and said second virtual machine implementer are operative to present said respective memory of said first computer and said respective memory of said second computer as a single shared memory to said guest operating system, and to distribute instructions of said guest operating system to said single shared memory.

43. The method according to claim 1, wherein said first and second computers comprise separate, respective first and second central processing units (CPUs), first and second memories, first and second I/O devices, and first and second buses that respectively interconnect the first CPU with the first I/O device and the second CPU with the second I/O device.

44. The computer software product according to claim 14, wherein said first and second computers comprise separate, respective first and second central processing units (CPUs), first and second memories, first and second I/O devices, and first and second buses that respectively interconnect the first CPU with the first I/O device and the second CPU with the second I/O device.

45. The computer system according to claim 28, wherein said first and second computers comprise separate, respective first and second central processing units (CPUs), first and second memories, first and second I/O devices, and first and second buses that respectively interconnect the first CPU with the first I/O device and the second CPU with the second I/O device.

APPENDIX B – EVIDENCE

1) Declaration under 37 CFR 1.132 by Dr. Joseph Landman and supporting exhibits, filed October 19, 2009.

2) Declaration by under 37 CFR 1.132 by Boaz Yehuda and supporting exhibit, filed October 19, 2009.

3) Declaration under 37 CFR 1.132 by Dr. Guy Tel-Zur and supporting exhibit, filed April 26, 2010.

4) Declaration under 37 CFR 1.132 by Dr. Dan Eylon, filed July 21, 2008.

All of the above declarations appear in PAIR on the respective dates.

APPENDIX C – RELATED PROCEEDINGS

None.